Санкт-Петербургский Национальный Исследовательский Университет Информационных Технологий, механики и оптики

Факультет инфокоммуникационных технологий

Лабораторная работа 4

«Проектирование инфокоммуникационных систем»

Выполнил: студент группы № К3420 Бейлин Максим Тимурович

Проверил: доцент ФИКТ Осипов Никита Алексеевич

Санкт-Петербург

2020

**Цель работы:**

* изучить и закрепить основы разработки диаграммы последовательностей;
* освоить IDE MS Visual Studio в части разработки диаграммы классов.

**Задачи:**

Разработать диаграмму последовательностей, показывающую взаимодействие объектов, упорядоченное по времени.

Разработать диаграмму классов на основе шаблонов для распределения обязанностей между классами.

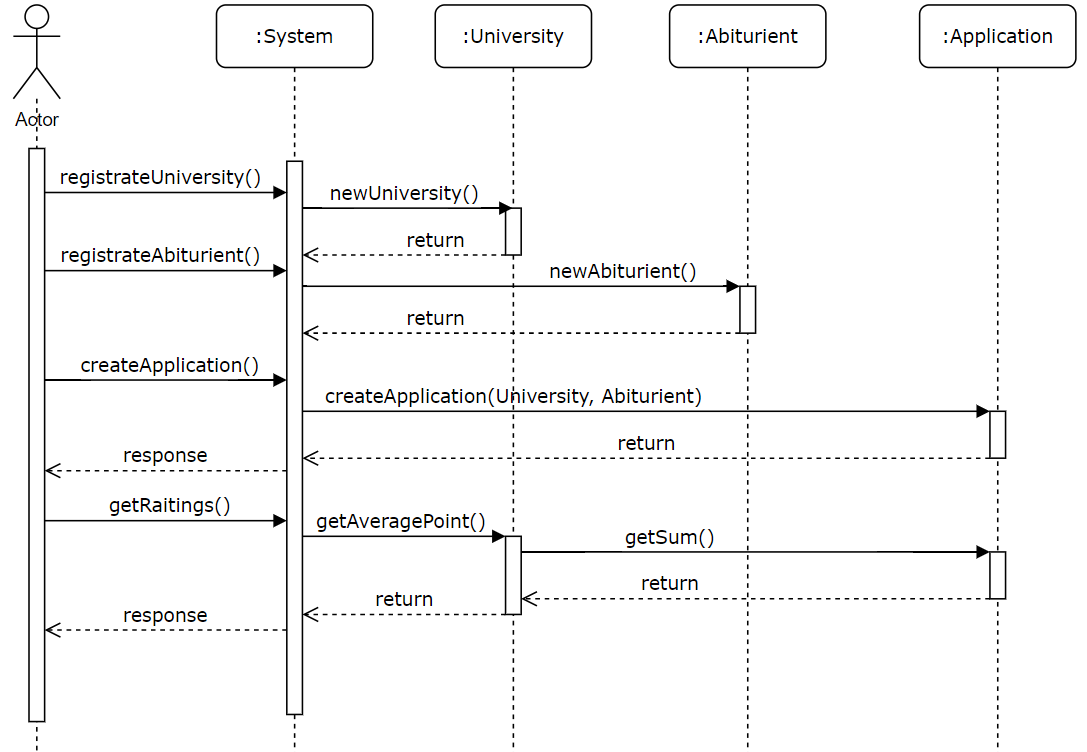
**Ход работы**

Первоочередным этапом разработки является описание видимых классов (сущностей), которое позволит в дальнейшем определить их зависимости друг от друга, а также построить диаграммы последовательности, т. к. их функциональность и зона ответственности уже будет частично определена в описании.

Основными классами являются:

* абитуриент (профиль абитуриента) – содержит всю информацию об абитуриента, в том числе личную информацию и данные, необходимые для подачи заявления на поступление в ВУЗ;
* ВУЗ (профиль ВУЗа) – содержит информацию об университете, об условиях поступления и проч.;
* заявление – содержит информацию о заявлении конкретного абитуриента на поступление в конкретный ВУЗ;
* контроллер (система, обработчик данных) – класс, содержащий информацию обо всех абитуриентах и ВУЗах и ответственный за работу с их выдачей (выборками).

На основе диаграмм DFD создается диаграмма последовательности, содержащая ту же последовательность выполнения задач (обработки данных) с учетом взаимодействия и существования во времени объектов описанных ранее классов:



1. *Диаграмма последовательности*

Основной проблемой при построении диаграммы оказалось определение зон ответственности классов. Так как система подразумевает обмен данными с внешним объектом (актором, пользователем системы), было принято решение о введении главного класса-контроллера, который одновременно хранит списки созданных объектов по ходу работы системы, а также делегирует поступающие извне задачи между ними.

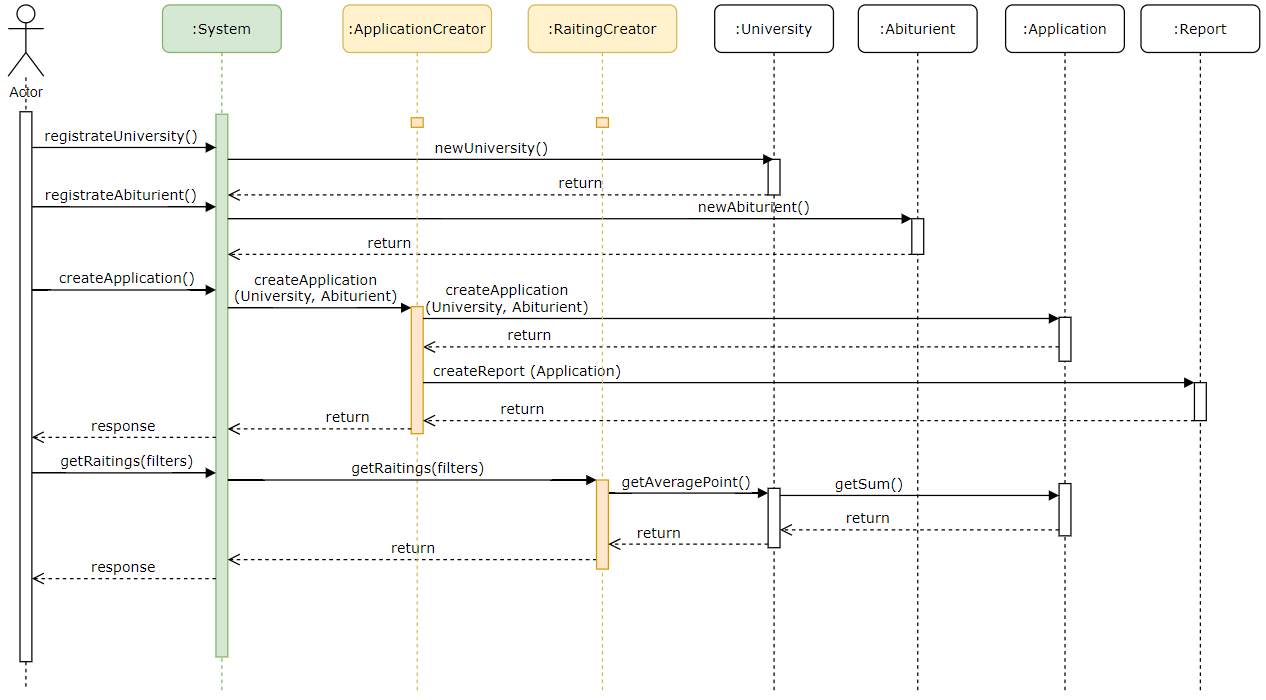
Наиболее важным элементом диаграммы является последовательность получения рейтингового списка университетов по запросу пользователя. Согласно делегированию обязанностей, выполнение данного запроса происходит поэтапно:

1. Главный контроллер для выполнения сортировки запрашивает у объектов-ВУЗов их средний бал при поступлении.
2. Каждый ВУЗ при этом вычисляет средний балл между поданными заявлениями, для чего запрашивает у объектов-заявлений эту сумму.
3. Каждый объект-заявление выполняет расчет суммы баллов, содержащихся у него в качестве полей.
4. Информация поэтапно возвращается «наверх», и главный контроллер осуществляет ответ пользователю.

Таким образом, в ходе проектирования были применены следующие шаблоны GRASP:

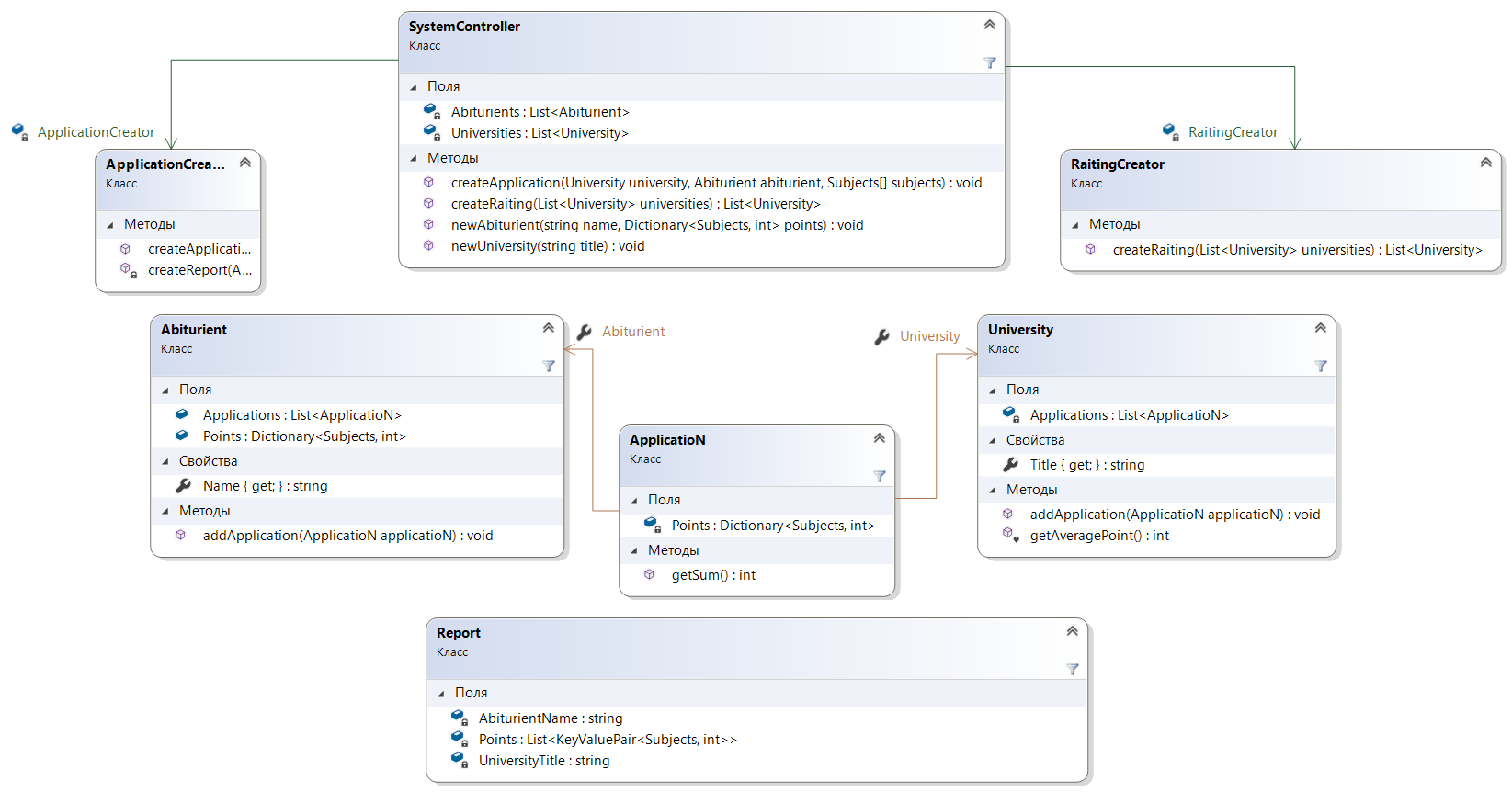
* шаблон Controller для класса System, так как он принимает обращения от актора и делегирует обязанности;
* шаблон Information Expert для класса University, так как этот класс обладает всеми данными для расчета среднего вступительного балла.

Однако, при использовании шаблона Creator возникла проблема, т.к. контроллер не может производить вычисления, а должен лишь перенаправлять запросы и возвращать ответы внешним вызовам. Было принято решение ввести в систему дополнительные классы-обработчики, которым, в соответствии с шаблонами Creator и High Cohesion были назначены функции создания объектов и работы с ними по принципу выполнения схожих и связанных по смыслу действий. В общем виде схема стала выглядеть так:



1. *Диаграмма последовательности, зеленый – контроллер, оранжевые – обработчики*

Теперь на основе диаграммы последовательности стало легче строить диаграмму классов, т. к. классы, их поля (свойства) и методы уже в основном определены. Диаграмма классов представлена на Рис. 3:



1. *Диаграмма классов*

В итоге структура проекта приняла вид из 7 классов:

1. Главный класс-контроллер системы SystemController **(шаблон Controller)** – класс, в задачи которого входит делегация всех внешних (пользовательских) запросов между другими объектами системы. В качестве полей содержит списки абитуриентов и ВУЗов (что может быть заменено на обращение к БД через отдельный обработчик с целью доступа к объектам). В рамках проекта рассматривает 4 основных вида запросов, им соответствуют следующие методы:
   1. Создание профиля абитуриента.
   2. Создание профиля ВУЗа.
   3. Создание заявления.
   4. Создание выборки-рейтинга ВУЗов.
2. Класс ApplicationCreator, создающий заявления **(шаблон Creator, High Cohesion)** – класс, отвечающий за регистрацию в системе заявления. Его главная задача – создавать объекты-заявления. Второстепенная задача – создавать отчеты о подаче заявлений для ведения учетности. Соответствующие методы:
   1. Создание заявления.
   2. Создание отчетности.
3. Класс RaitingCreator, создающий выборку ВУЗов **(шаблон Creator)** – класс, отвечающий за создание выборок по ключам. Его основной задачей и единственным методом является метод получения выборки, т. е. отсортированного списка подходящих ВУЗов.
4. Класс Abiturient, отвечающий за профиль абитуриента. В качестве полей и свойств может содержать такую информацию, как: имя, баллы ЕГЭ, список поданных заявлений. Методами класса являются методы добавления информации и заявлений.
5. Класс University, отвечающий за профиль ВУЗа. Содержит информацию о самом ВУЗе, а также о списке поданных заявлений. Из методов:
   1. Добавление нового заявления.
   2. Получение среднего балла поступающих.
6. Класс Application, описывающий заявление абитуриента. Содержит в качестве поля ряд пар предмет-баллы, соответствующие результатам ЕГЭ абитуриента. Единственным методом (по шаблону Information Expert) является метод для расчета суммы результатов ЕГЭ.
7. Класс Report, описывающий отчет о поданном заявлении. Содержит в качестве полей имя абитуриента, название университета и список баллов.

**Выводы**

Основной частью работы стало изучение принципов шаблонов GRASP, т. к. при первоначальном подходе к построению диаграмм возникли достаточно серьезные трудности в определении ответственности классов и распределении обязанностей между ними. Так, наибольшей сложностью отличилась задача выбора классов-создателей объектов. На данном этапе проектирования было принято решение разгрузить главный класс приложения на несколько создателей, занимающихся каждый объектами своих классов. Так появились отдельные классы-обработчики со своей зоной ответственности.

Построение диаграммы классов в IDE Visual Studio оказалось удобным и практичным с точки зрения воплощения проекта в виде программной его части процессом. Во время графического построения диаграммы параллельно происходило написание программного каркаса проекта, что позволило сэкономить время на написание кода. Единственным минусом, замеченным во время выполнения работы, оказалось отображение зависимостей между классами на диаграмме. Так, отношения ассоциации отображается лишь когда полем является объект данного класса, но не список (List).

В итоге, по завершению данной работы был готов программный каркас проекта, и для завершения написания прототипа остается наполнить методы бизнес-логикой и настроить минимальный интерфейс для взаимодействия с приложением.